Arduino Nano as IR sender and receiver, part 2: details

Table of contents

Date	Description
2016-01-08	Initial version.
2016-01-11	Added section for symlink-broken systems.

Table 1: Revision history

1 Introduction

This document is a supplement to the first part. It contains some additional information, left out of the first part for the ease of accessibility.

A third part is planned, elaborating on further possibilities of AGirs, and showing some hardware solutions, in particular on more capable boards than the ATMega328 of Ardiuino Nano/Uno.

2 The AGirs Software

Girs is a specification for a modular command language for command language of a server, capable of sending and receiving IR commands, and more. "Modular" means that an implementation can select whether to implement different "subsets" (called modules) or not. AGirs is a project that implements a subset of the Girs specification on the Arduino platform. It is highly configurable using C pre-processor symbols and a configuration file. *GirsLite* is a special configuration of AGirs, aimed at IrScrutinizer and Lirc. Previously, there existed another configuration called *Girs4Lirc*, aimed specifically at Lirc, but it has now been merged into GirsLite.

The AGirs package allows a lot of other nice things, like sending and decoding some IR protocols directly (presently only NEC1 and RC5), support for LCD display and a number of signaling LEDs, named commands (a data base (like Lirc) of commands identified by names like `Play`), support for sending RF commands, Ethernet (TCP and UDP sockets) support (instead of using the serial port). See the documentation in the project.

AGirs is not a finished project. Cooperation is welcome.

3 Alternative Components

There is nothing magic about the selection I have made in part 1. Many different alternatives exist, which will be discussed next.

The Nano board and the ATMega328 are hardly "hot" any longer. However, its performance and memory configuration is more than enough for simple applications, like in IrScrutinizer and Lirc. If considering "hotter" alternatives, note that ATMega328 boards can supply (up to) 40mA on the GPIO pins, more than enough to shoot an IR signal with one or two high-performance LEDs at quite a distance. The modern 32bit processors typically are specified for 7mA, making a driver transistor, (like 2N7000/ BS170) necessary.

The main reason for the selection of the Osram IR LEDs is that the black packaging looks cool :-). Many other alternatives, for example the Vishay TSAL6100 (narrow beam) and the TSAL6200 (wide beam) exist. However, be sure to use IR LEDs with a wavelength of 940-950nm, corresponding to Consumer IR, not a 890nm component, intended for use with IRDA. An IR LED of the latter type will almost surely "work", but with non-optimal performance.

Of course, using only one LED is also an option. Be sure to check the resistor value, see the Appendix.

(Some) alternatives to the TSMP58000 are TSMP58138, TSMP1138, and TSMP4138 (the latter is found to the right on the IrScrutinizer splash screen). They all have similar properties, very good amplification, but ends at around 60kHz modulation frequency. The smaller sensors, QSE157 (used in the IrWidget, now end-of-life and replaced by OPL551), QSE159 (used in IrToy), Honeywell SDP8610 can also be used, but due to its smaller effective sensor area, makes the sensitive range smaller. Also, the "small" ones in general do not run on 3.3 Volts.

> Note:
>
> Note that the different chips are in general not pin compatible. Bu sure to check the data sheet.

All of the sensors mentioned have inverting output, which is what the firmware expects. There are also sensor with non-inverting output available. These are not directly supported.

The experiments I have made with photo diodes (potentially being able to handle much higher modulation frequencies) show that the range reduces to millimeters.

Alternative IR Receivers: First, note that the last two digits of the number indicates the modulation frequency. For a receiver, this is to be understood as the center frequency of a band-pass filter. An IR receiver marked "38kHz" will work quite well for modulation frequencies between 36 and 40kHz (at least). (See e.g. Figure 5 in the data sheet for TSOP34438). As far as I know, all "universal" IR hardware, IrToy, Iguana, TIRA, USB/ UIT, CommandIR,... come with 38kHz receivers. However, if searching the optimal solution for a certain remote or protocol etc, getting the optimal receiver tuned to the actual modulation frequency is definitely not a bad idea.

4 Testing the firmware

When the sketch is started, as a self test, the (visible light-) LEDs, and the "pin13"-LED, are turned on for two second. This is the first test, requiring no additional hard- or software.

When running the GirsLite sketch, the board is nothing but a server in the sense of the Girs Command Language. It can be accesses by programs like IrScrutinizer and Lirc (using its Girs driver), but also by humans using a terminal program, like the serial monitor of the Arduino IDE. For this, set the baud rate to 115200, and the line ending to carriage return. It is now possible to communicate with the unit using the terminal program. Just type the command to the program, and the unit will respond. One line in, one (possibly long) line out. GirsLite supports the commands `modules`, `version`, `transmit`, `analyze`, `receive`, and `led`. All commands can be abbreviated to the first letter. For example, to test the `receive` command, just type "r" (without the quotes), followed by return, and fire a suitable IR signal at the receiver. The raw capture will be output to the terminal program. Using the clipboard, it can be pasted to IrScrutinizer, and analyzed. Also the other commands can be tested in this way.

5 Compiling the sources

First install the latest Arduino IDE (at the time of this writing, this is 1.6.7) from arduino.cc. (Do not use "1.7.x" from `arduino.org`.)

Some proficiency using the Arduino IDE is required. This is covered at the Arduino site.

Download and unpack the AGirs software. Copy, move, or link `src/GirsLib` therein to your Arduino libraries folder. Then download and unpack Infrared4Arduino repository into your Arduino libraries folder.

As is common in the Open Source/Git community, this project contains symbolic links to organize the sources and prevent duplication. This usage is not discouraged in any document I have seen so far, neither in the Git handbook nor in the Github texts. However, on Windows this can cause problems, at least for some Git implementations. Long story short: if `src\GirsLite\GirsLite.cpp` just contains the line `../Girs/Girs.cpp`, then delete the file and replace it by the file `src\Girs\Girs.cpp`.

Now connect the board and start the Arduino IDE on the file `src/GirsLite/GirsLite.ino`. In `Tools -> Boards` select `Arduino Nano`. In `Tools -> Processor` select `ATmega328`. In `Tools -> Port` select the port the board is currently connected to (`COMN` (for some N) on Windows, `/dev/ttyUSB0` or similar with Linux,...). Pressing the "upload" button will now (hopefully!) compile the program and upload ("flash") it onto the board.

6 Appendix. Computing the current through the IR LEDs

Given *n* IR LEDs connected in series between ground and a resistor *R* to the voltage *V*. Compute the current I through the components! For this, first *guess* the answer I (say 40mA, which is the maximal current from the AtMega328 GPIO pins). Then use the data sheets for the IR LEDs to determine the forward voltage over the *i*-th IR LEDs, V_i (typically 1.4V). So the voltage over *R* is thus $V - (V_1 + ... + V_n)$, and, by Ohm's law $I = (V - (V_1 + ... + V_n))/R$. Now go back and check that the original guess was reasonable, and repeat it necessary. For *V*, this is clearly "somewhat" lower than Vcc, see Figure 29-163 in the ATMega328 datasheet. 4.5V might be a reasonable value.

From a practical point of view, using the standard USB as power, it seems reasonable to simplify the formulas as $I = (4.5 - n\ 1.4)/R$.

7 Appendix. Configuration of udev for Linux

Linux (the subsystem `udev` really) assigns a connected Arduino a device file name of the form `/dev/ttyACMn`, in some cases `/dev/ttyUSBn`, where `n` is the smallest non-negative integer not yet taken. This can cause unpredictable behavior, not only when using several Arduinos, but also in context of other device using the same names, like IrToys. By using custom rules to udev this difficulty can in general be circumvented.

Since there are so many different manufacturers of "Arduino-compatible" hardware, "all" having different vendor id and product id, there is no single simple answer. Some comes with a unique serial, cheaper clones in general not. As a guide to the reader, not as a definite answer, this is my `/etc/udev/rules.d/10-arduino.rules`.

```
SUBSYSTEMS=="usb", ATTRS{idProduct}=="0043", ATTRS{idVendor}=="2341", SYMLINK+="arduino
 arduino_uno arduino_uno_$attr{serial}"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="7523", ATTRS{idVendor}=="1a86", SYMLINK+="arduino
 arduino_nano_qinheng"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="2303", ATTRS{idVendor}=="067b", SYMLINK+="arduino
 arduino_nano_prolific"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="6001", ATTRS{idVendor}=="0403", SYMLINK+="arduino
 arduino_nano_ftdi arduino_nano_ftdi_$attr{serial}"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="8037", ATTRS{idVendor}=="2341", SYMLINK+="arduino
 arduino_micro"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="8036", ATTRS{idVendor}=="2341", SYMLINK+="arduino
 arduino_leonardo"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="0042", ATTRS{idVendor}=="2341", SYMLINK+="arduino
 arduino_mega arduino_mega_$attr{serial}"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="804f", ATTRS{idVendor}=="2a03", SYMLINK
+="arduino.org arduino_m0_pro"
SUBSYSTEMS=="usb", ATTRS{idProduct}=="2111", ATTRS{idVendor}=="03eb", SYMLINK
+="arduino_m0_pro_prog"
```

Note that these are not "real" names but symbolical links, like extra, alternative names. They are not really usable for the present version of IrScrutinizer. For Lirc, it can be useful however.

8 References

1. Reference documentation for the Arduino Nano board. For clones (which are in general based on the open-sourced Eagle files), this is also in general quite reliable, the main difference is that the clones do not use a FTDI chip but a CH340 chip.
2. Schematics for the Arduino Nano board. Same remark as above.
3. Data sheet for ATmega328 (and some others).
4. IR LEDs Osram SFH4544, IR LED with narrow beam (half angle ± 10°), data sheet.

5. IR LEDs Osram SFH4546, IR LED with wide beam angle (half angle ± 20°), data sheet.
6. Non-demodulating IR sensor TSMP8000, data sheet.
7. Demodulating IR Receiver TSOP34438, data sheet.